Alexander Prosser and Reinhard Steininger

e-voting2006.at - An Electronic Voting Test Among Austrians Abroad

Working Paper

e-voting2006.at

An Electronic Voting Test Among Austrians Abroad

Alexander Prosser, Reinhard Steininger

Arbeitspapiere zum Tätigkeitsfeld
Informationsverarbeitung und Informationswirtschaft
Working Papers on
Information Processing and Information Management

Nr./No. 02/2006

Herausgeber / Editor:
Institut für Informationsverarbeitung und Informationswirtschaft
Wirtschaftsuniversität Wien · Augasse 2-6 · 1090 Wien
Institute of Information Processing and Information Management
Vienna University of Economics and Business Administration
Augasse 2-6 · 1090 Vienna

e-Mail: alexander.prosser@wu-wien.ac.at
WWW: http://e-voting.at

Table of Content

Preface

Electronic citizen participation has become a realistic option on all levels. Electronic participation includes: (i) citizen information systems about political decision making and law making, such as parlinkom.gv.at; (ii) discussion and deliberation platforms; and (iii) direct decision making in electronic voting, which is the focus of this research project.

The high level of international experience in the field of electronic voting has been encouraging. In a seminal contribution, the Council of Europe published a set of minimum requirements for the legal, operational and technical design of electronic voting systems [CoE2004]. There is an ever-increasing number of pilot projects been conducted in several European countries.

Practical experience is needed, not only to test the technology, but also to test the usability and user acceptance of such systems. This was the main objective of this test.

We would like to take this opportunity to thank all those, who made this test possible:

Our main sponsor, Wiener Zeitung GmbH, the Official Journal of the Republic of Austria, and its general manager, Mag. Karl Schiessl.

The IT Department of Wiener Zeitung, which was responsible for hosting this test, in particular Mr. Josef Berger, Mr. Tilfried Weissenberger and Mr. Michael Kick as well as the Marketing Department, in particular Ms. Nadja Traxler-Gehrlich for the excellent cooperation in preparation and during the test.

The “election committee” supervising the opening and counting of the ballot, Prof. Gabriele Kotsis (President of the Austrian Computer Society), Dr. Kurt Breitenstein (Vice Dean of the University of Economics and Business Administration) and Dr. Andreas Unterberger (Editor in chief of Wiener Zeitung).

Many thanks go to the guest speakers at the symposium on e-voting held on the 14th of October:

Dr. Nadja Braun (Swiss Federal Chancellery),
Dr. Michael Remmert (Council of Europe) and Mr. Norbert Rzesnik (Ministry for Social Affairs, Germany).

The Report is dedicated to these persons.

The authors, December 2006

1 The Aims of the Test

This was the third field test conducted with a prototype from the research initiative e-voting.at at the University of Economics and Business Administration, Vienna (WU). The test had already been conducted in 2003 (parallel to the Student Union Election, 300 participants) and 2004 (parallel to the Presidential Elections, 1700 participants) [PKK03, WU2004]. However, these tests had been conducted among students of WU (2003 even only among students of business computing), this time, the test addressed the general public.

Particular emphasis was hence placed on the usability of the prototype and on the feedback collected in a usability lab as well as from a questionnaire that was offered to the user after vote casting.

For each of the tests mentioned, a newly developed prototype was used, exploring different technology and user options. In this case, the new options explored were:

The **recovery of the voting card** (see Section 2 for the process implemented), where the card could be recovered by the user in case of loss. This would be a considerable improvement as compared to postal voting procedures, where lost voting documents may not be reproducible.

Preferential votes or sub-options in the ballot, where the voter could first select from a main option and depending on his choice a number of sub-options further defining the main option were offered for selection (see also copy of the ballot sheet in Fig. 6).

The ballot is encoded with the public keys of the election committee to prevent fraudulent manipulations by the election system administration. The procedure is that the private keys remain with the respective committee member until vote casting terminates and the ballots are to be opened and counted. In the previous tests, the private keys literally stayed with the **committee members**, where they could be lost. In this test a **Key Store** was used to generate RSA key pairs, whereby the private keys remained in the Key Store and could only be retrieved by the committee members themselves.

One of the key requirements for a secure electronic voting system is the information separation between client and server. Some information must not be known to the server. An example is the encryption of the ballot before it is sent to the electronic ballot box. Encryption prevents the server administration (or whoever has access to the election server) from being able to read the ballots. However, if the encryption is done on the server, it is literally useless, as the server would “see” the ballot in its unencrypted form. Therefore, decentral logic is needed in the voter’s Web browser in order to encrypt the ballot sheet before it is sent to the server and it was implemented in this test as a Java applet. Java applets require a Java run time environment (RTE), which need not necessarily be installed on a PC. The **user guidance** in ascertaining whether a **Java RTE** was already installed on the PC, whether it was the right version (or higher) and, if not, to download the Java RTE was another focus of this project.

In order to gauge the **user’s perception of the voting process**, a questionnaire was added after vote casting.

2 The Process Used

The entire voting process is split into two parts: (i) voter identification; and (ii) vote casting. This is necessary to maintain anonymity.

Identification

The prototype developed at the University of Economics and Business Administration Vienna (WU) requires user authentication via login (also digital signatures had been tested, see [PKK03]), however as this test addressed Austrians abroad and since no voter register was available (this was an unofficial test), authentication was based on self-registration. This screen, as was the prototype itself, was implemented as a Java applet. The screen that collected names and addresses of the participants (R1, R2 in Fig. 1) was added as a separate module and the original login screen was disabled. In conformance to the Austrian Data Protection Act (DatenschutzG 2000), user data was deleted after the analysis of the test, unless the user, upon registration, had indicated his consent for the data to be kept and to receive a copy of this report by email.

The second screen requested the user to enter and confirm the password that was used to symmetrically encrypt the voting token (R3). A file dialogue box prompted the user to specify the file name and location of the voting token to be stored locally (R4). The voting token was a large random number (due to the size of the number, uniqueness could be assumed) generated by the applet. In this screen, the user could also indicate, whether he wanted to use the recovery function for the voting token (see below).

The token was sent to the election server and the verifier to be blindly signed by both[1]. In this test neither server checked against a voter register, as the test was based on self-registration, hence, any incoming request to authenticate a token was accepted and performed by the election server and verifier (R5a,b).

The blindly signed tokens were sent back to the client and unblinded (R6,7); as a result, the client possessed two authentic signatures on its random token number. The tokens were formatted in an XML structure, which was symmetrically encoded with the password indicated by the user in R3. The resulting file was stored locally at the file location selected by the user in R4 (R8).

If the user selected the recovery option, the password-encoded token was sent to the server to be stored with the logged authentication request (R9, R10). If the user lost the token, he could have requested the token to be resent by email. Since the *encoded* token was stored, neither the election server administration nor anybody intercepting the response email in the case of token recovery, would be able to open and abuse the token. Only the person with the correct password was able to use the token. On the downside, not even the system administration could help the user recover the token, if the password was lost.

The identification process concluded with the respective confirmation message, which again reminded the voter of the file location selected for the voting token. (R12). All steps contained error messages pertaining to user as well as system errors. An error was also displayed on the client's PC if communication with the server was not possible within a certain time limit.

Election Committee

One of the main issues in any e-voting system is the system administration's ability to manipulate votes. An appropriate way to prevent such manipulation is to decentrally encrypt the ballots (at the voter's PC) with the public keys of the election committee before they are sent to the e-voting server. Only after vote casting, when the members of the election committee have provided their private keys, can the ballots be opened and counted.

In this test, the election committee consisted of three members. The process of requesting the election committee key for one member is shown in Fig. 2. This process must take place before vote casting starts.

Each committee member sends a digitally signed request to a Secure Key Store to generate an asymmetric key pair (P1, 2). The traffic between client and Key

[1] Blind signatures enable a requestor to obtain an authentic signature from a signor without the signor knowing what it signed (for details see [Cha82]).

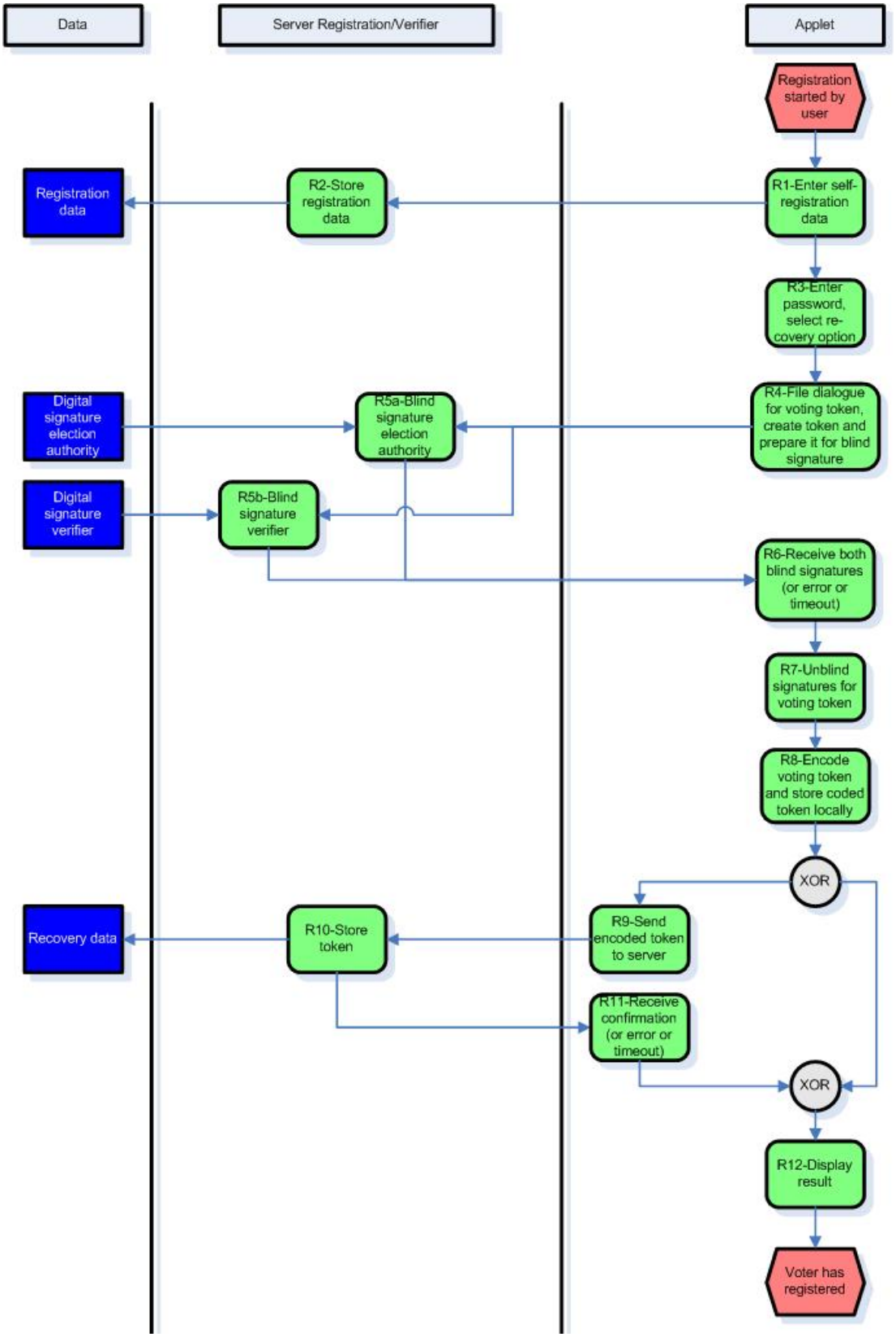

Figure 1: Registration process

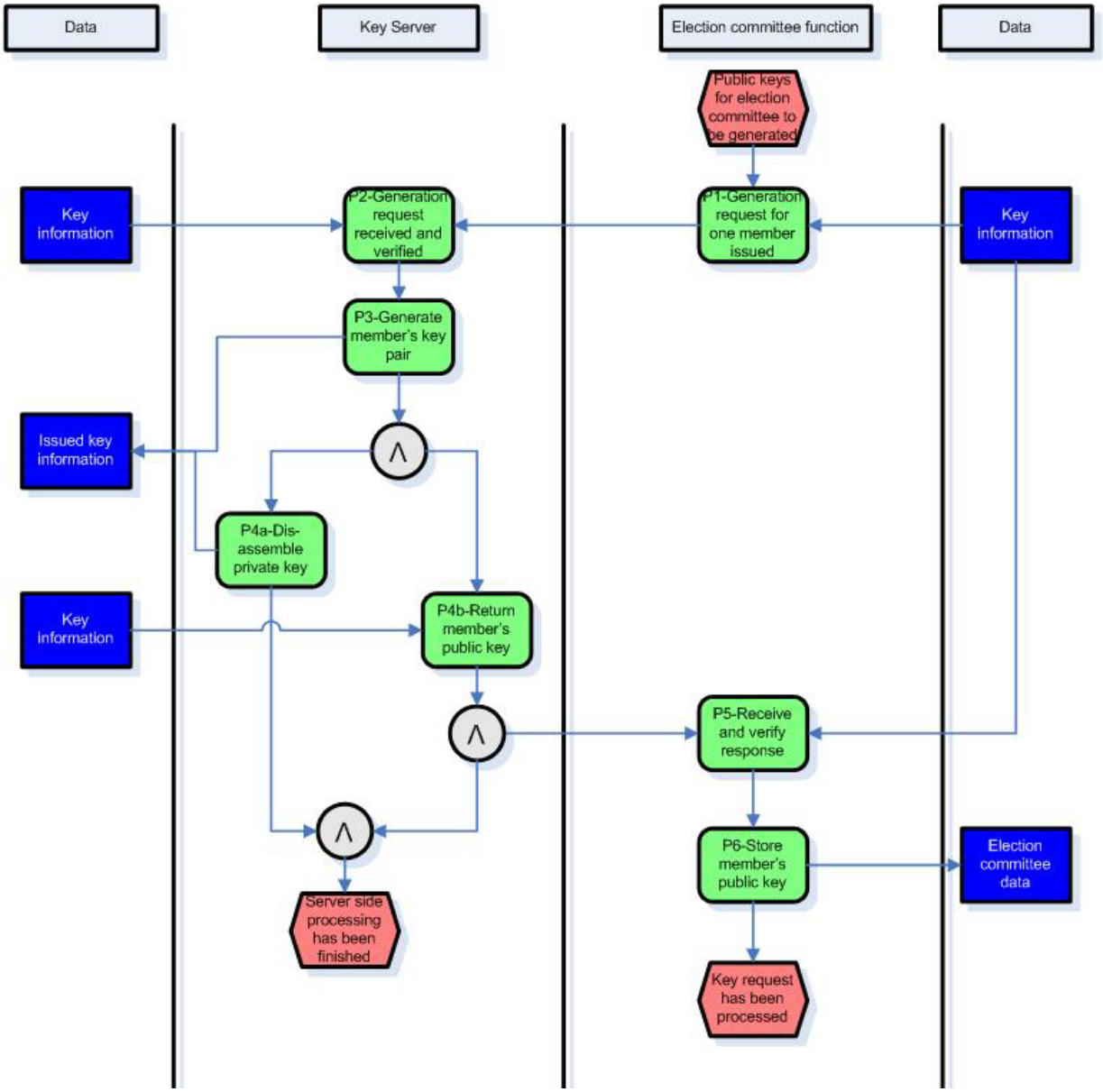

Figure 2: Public key generation for committee members

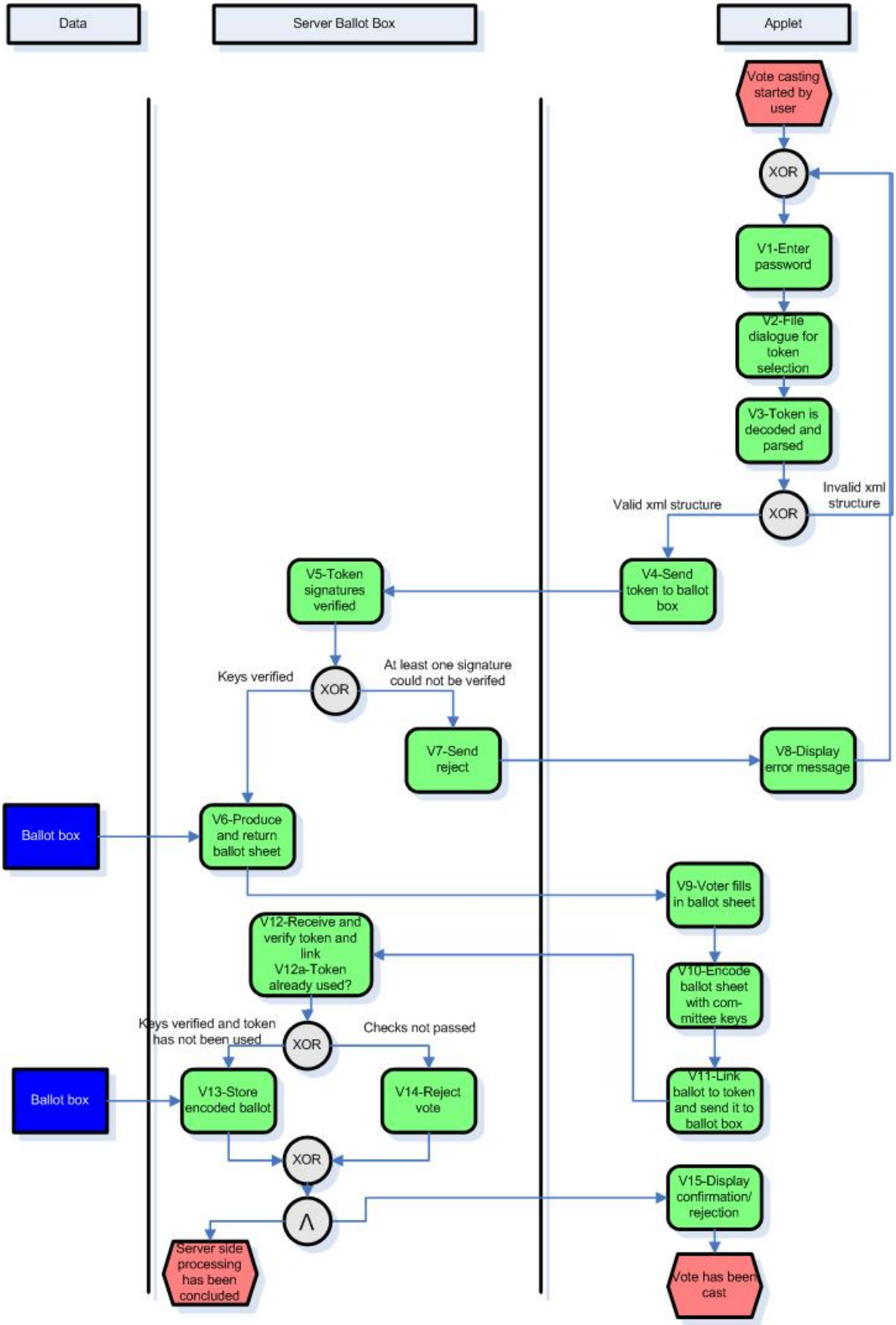

Figure 3: Voting process

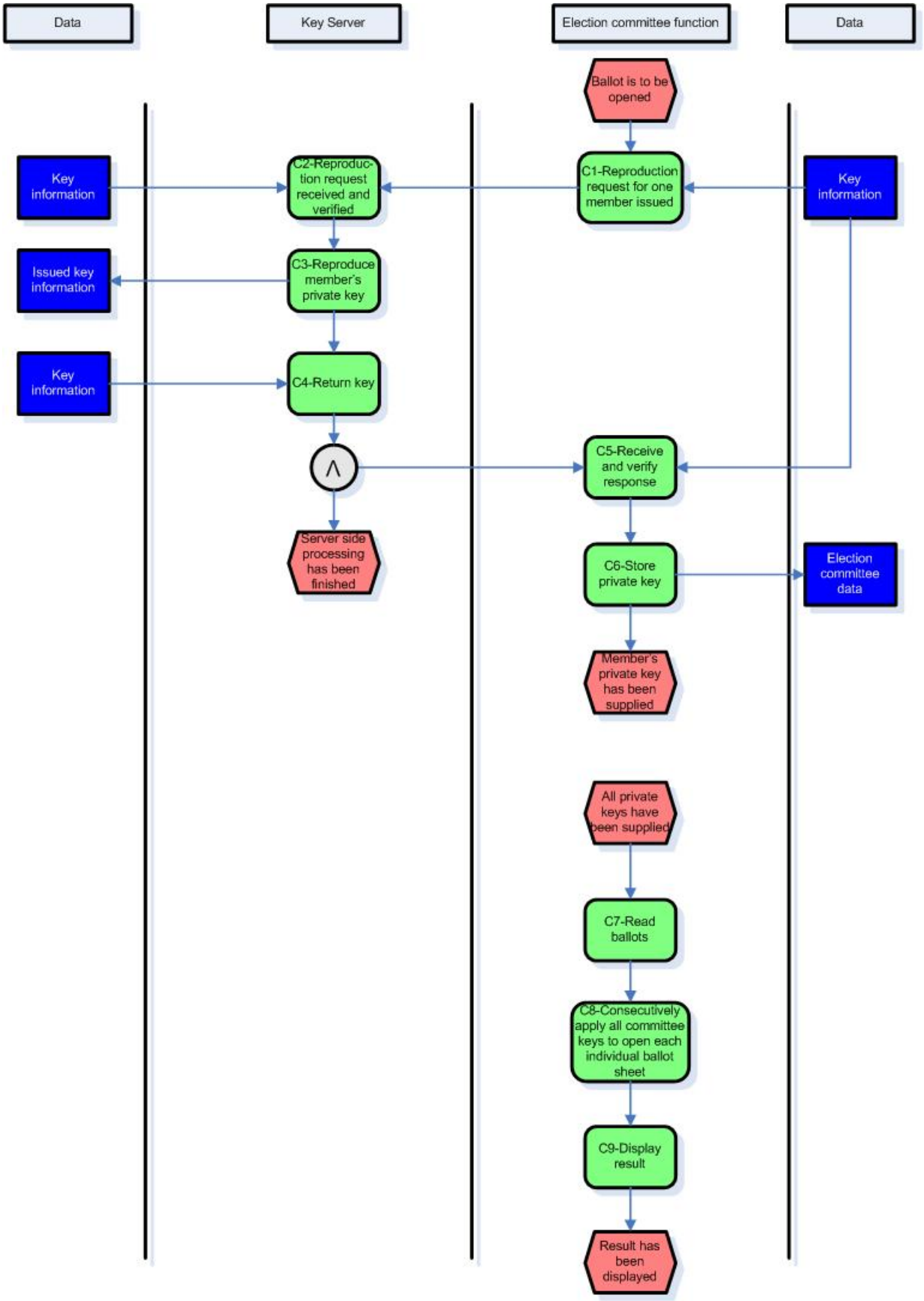

Figure 4: Opening of the ballot

Store is 1024 bit encrypted and authenticated using RSA keys[2].

The public keys of the committee members are returned to the election committee server and are later delivered to the voter together with the ballot sheet. However, the private keys remain in the Secure Key Store, inhibiting unauthorized access. They can only be retrieved by a digitally signed request of the original requestor. It is only with these private keys that the ballots can be opened, preventing both the administration and/or a single member of the election committee from gaining access. This safeguard ensures that only authorized members of the election committee may *jointly* open the ballots.

The issue of the request in the election committee system and its reception in the Key Store as well as the respective response are logged in a tamper-proof encoded log file.

Voting

The voter was guided though the authentication and voting process with a second Java client applet. The voter was first required to enter the password used to encrypt the voting token and to specify the location of the file containing the token (V1,2). The token was then decoded with the voter's personal password and it was attempted to parse the XML structure (V3), the attempt failed, if the user indicated an invalid or corrupted token or an invalid password. In this case, the user was prompted again for the password and token file.

If the XML structure of the token was parsed successfully, the signed tokens (election authority and verifier) were sent to the ballot box to be verified (V4,5). If only one of the signatures could not be verified, the request was rejected (V7,8), otherwise the ballot sheet was returned together with the keys of the election committee members (V6). The voter filled in the ballot sheet and before the ballot was cast, it was consecutively encoded with the keys of the election committee members. The encoded ballot sheet, inextricable linked to the token was then sent back to the ballot box server (V9-11).

The ballot box proceeded once again to verify the authenticity of the token and the link between token and encoded ballot (V10). The ballot box then either stored the encoded vote or rejected it (V13-15).

Opening and counting of the ballot

After vote casting had finished, the members of the election committee provided their respective private keys by issuing a request to the Key Store to reproduce and provide the private keys that correspond to the public keys, which had been used to encrypt the votes (C1).

The request was once again encrypted and verified with 1024 bit RSA keys. After verifying the requestor, the Key Store reproduced and delivered the private key for the respective committee members (C2-4). The upper part of the process depicted in Fig. 4 shows the procedure for one individual committee member.

The election committee function receives the keys and stores them for future use (C5,6). Once all private keys have been provided, the ballot can be opened and counted (C7-9).

[2] For an introduction, see [Sch01].

3 Security Analysis of the Process

In 2004, the Council of Europe published a set of minimum legal, operational and technical requirements for voting systems [CoE2004]. Its Appendix III lists the possible threats to which a voting system exposed. Below, the voting process is discussed according to the criteria set (headings below) by the Council of Europe.

T.Audit_Forgery and T.Observ_Forgery

This concerns the forgery of audit data collected. All log files used are RSA encoded with the private key stored externally and physically separated from the operational system.

T.Auth_Disclosure

This concerns the disclosure of login and authentication data enabling impersonation of the respective person:
1. Voter authentication in identification: N/A because the process was based on self-registration.
2. Voter authentication in vote casting: The voter is authenticated according to a voting token, which only the voter himself may open. As long as the voter keeps the password secret, nobody is able to use said token.
3. Login to the election committee software and Secure Key Store: Logins are stored as standard SHA-1 password hashes and Web service requests are RSA-authenticated.

T.Hack

This requirement is sometimes misinterpreted as primarily concerning conventional server security. This however, is a fundamental misconception because: (i) servers may always be vulnerable, hence a voting system whose security is solely based upon server security may be questioned in general, (ii) the most dangerous attacker of all, the system administrator of the election system, always has access to the servers and hence, the necessary implementation of additional safeguards such as ensuring that: (A) even the system administrator may not match the vote with a voter (see T.Vote_Confidentiality); (B) even the system administrator may not abuse a voting token (even when stored in the recovery function) to impersonate an eligible voter and to steal his vote (see T.Vote_Impers); (C) even the system administrator cannot modify votes cast (see T.Vote_Modify); and (D) even the system administrator cannot insert forged votes (see T.Vote_Impers).

T.System_Forgery

This concerns redirecting a Web page to a fake website in an effort to mislead the voter into believing that he had indeed voted. The standard measures are to digitally sign the applets used. However, since this was an academic test and considering the costs incurred for such a signature, self-signed certificates were used for the Java applets. The main site, however, had a valid https certificate.

T.CandList_Disclosure and T.CandList_Modify

Since the list of candidates (that is options to choose from) were hard-coded in the prototype used, all issues concerned with lists stored in a database, therefore did not apply. In most referenda and elections in Austria, candidate (or options) disclosure is not an issue, as the available choices are published well before the vote. Also in this test, the list of options were known beforehand (not least because screen cam shows were made available to users beforehand). List modification, however, becomes an issue as soon as variable candidate lists are sent from an election server and rendered by the client. In this case, protection needs to be provided in order to prevent tampering with the ballot sheet, communicated to the client.

T.Malfunction_XXX

This threat is defined for several stages in the election (hence summarized here as “XXX”) and it concerns the deletion of data used. This is primarily an operational issue and in this test, all servers concerned were mirrored (see Section 4) and therefore the deletion of data would have been detected.

T.XXX_DOS

This concerns threats through Denial of Service (DoS) attacks and also applies to all stages of the election. The effective protection from DoS attacks is primarily an infrastructure question and does not concern the process design as such.

T.XXX_Time

Many processes in an election are time-specific. An example may be a voter trying to cast a vote just before

the closure of the ballot box. Options to fulfill this requirement include linking the servers to an external time service (the option selected in this case) or the usage of an external timestamp service.

T.Privacy and T.Voter_Privacy

This concerns the disclosure of voter's and/or candidate's private data. E-voting2006.at did not store any candidate data and due to the self-registration process the only data stored was that indicated by the voters themselves.

T.VoteReg_Disclosure and T.VoteReg_Modify

This concerns disclosure of data from the voter register and did not apply, as no register was used and the respective parts of the prototype had been disabled in the deployment.

T.Ballot_Forgery

This concerns the fraudulent modification of the ballot sheet displayed, see also T.CandList_Modify.

T.CommD_Avail_pre

This concerns the availability and integrity of the data from the pre-voting stage. As the Recommendation mentions, this data is not required if voting is based on an anonymous voting token, which is the case in this system.

T.CommDSec_pre

This concerns the confidentiality of the voters' register. Again, as specified in the Recommendation, this requirement does not apply to situations (rather it is "automatically" fulfilled), where the right to vote is vested in an anonymous voting token.

T.Vote_Confidentiality and T.Vote_Trail

This concerns the ability of an attacker to match the voter with the voter's actual vote. Several lines of attack which have to be considered are:
1. When the verifier or the election authority sign a token request, they know, which voter issued the request. However, the blind signature prevents them from tracing the token itself back to the voter. The processing of the token itself (generation and blinding R4, unblinding in R7, creating the XML structure and encoding the token after the signatures, R6-R8) was done *decentrally* without the server having access to the data processed.
2. When the token is used to obtain a ballot and to cast a vote, it does not contain any data that could be used to trace it back to the voter.
3. The tracing of the IP address in the registration and vote casting process in order to map the voter and to the corresponding vote. Due to the clear technical separation of these two phases, this mapping is not possible. The registration server may log IP addresses of registrants but does not "see" the voting token issued. The same applies to the ballot box, which may also log IP addresses, but does not know (i) who is "behind" the voting token and (ii) the ballot is stored in the ballot box encrypted and only the committee members can open it.; in addition, the opened ballots are not stored persistently and every recount requires to go back to the original encoded ballots.
4. Intercept the ballot sheet when it is sent to the ballot box. Since the ballot is decentrally encrypted at the voter's PC, this attack will yield no result.

T.Vote_Modify

This concerns the ability to intercept a vote and to meaningfully change it. Since the vote is encoded with the keys of the committee members neither the system administrator nor a third party may meaningfully change votes. The only option left to an attacker gaining access to the voting data by whatever means is to aimlessly delete votes, which would, however, be at least noticed by comparing the tamper-proof encoded log files to the votes in the database after the election.

T.Vote_Multiple

This concerns protection from voters casting multiple votes either via e-voting or via multiple channels. Multiple e-votes are prevented by the check in Step V12a. Multiple voting channels were not used in this test.

T.Vote_Impers

This concerns impersonating an eligible voter to cast a vote (and to effectively steal the vote). The following possibilities exist in the process:
1. A system administrator of the election authority or the verifier or a third party intercept the token signature request. However, since the blinding parameters are only known to the client, the attacker could not extract the net token without the blinding layer (see [Cha82]).
2. Stealing the voting token either from the voters file or backup system, an attack during data communication

or (as a system administrator) from the token recovery database. This, of course, may occur at any time, but the attacker would not be able to decode the voting token as this requires the personal password only known to the legitimate voter.

T.CommD_Avail_elec

This concerns availability and integrity of data from the voting stage. It requires standard infrastructure measures and as far as attacks on the data are concerned, the argument follows T.Vote_Multiple.

T.CommD_Sec_elec

This concerns the confidentiality of data from the voting stage. Nobody can read the votes until the committee members provided their private keys. Even when the ballots are opened to be counted, the system does not store the decoded votes in the database. Every recount is done with the original encoded ballots.

T.MisCount and T.Result_Modify, T.Report_Modify

This concerns protection from incorrect counting. The appropriate protection would be to store votes in two independent electronic ballot boxes and to initiate an independent count to see, whether results tally. The prototype being used for this test did not accommodate for this option. However, its implementation on top of the existing process is rather straightforward. Another option would be to manipulate the coding used (which is a general issue). In its simplest form it would consist of the insertion of a statement that, with x% likelihood, deletes votes for a certain party or referendum option. In its general form this could be considered as threat T.ElectionSoftware_Modify. This threat can only be countered by software certification that involves the source code as well (i.e. Common Criteria using an EAL level or augmentation that involves direct source code inspection, see [CC06]) combined with digitally signed code used in deployment. In this case, all parties concerned can be assured that the software being used is actually the software inspected by the certification authority. Considering the costs of such a certification inspection, it is obvious that such measures were not undertaken for this test.

T.Partial_Count and T.Premature_Count

This concerns an attack, whereby partial results are computed from the votes cast in order to track voters or to illicitly obtain early voting results. This was prevented by the fact that the encoded votes could only be jointly opened by the election committee.

T.Vote_Duplicates

This concerns the insertion of duplicate votes either due to an attacker (in this case it would be a subset of T.Vote-Multiple, see above) or due to malfunction. The latter was prevented by storing the voting token together with the vote cast as well as the data inextricably linking them together. Since a unique index was used for the database field storing the token, no token could have been entered twice.

4 Implementation

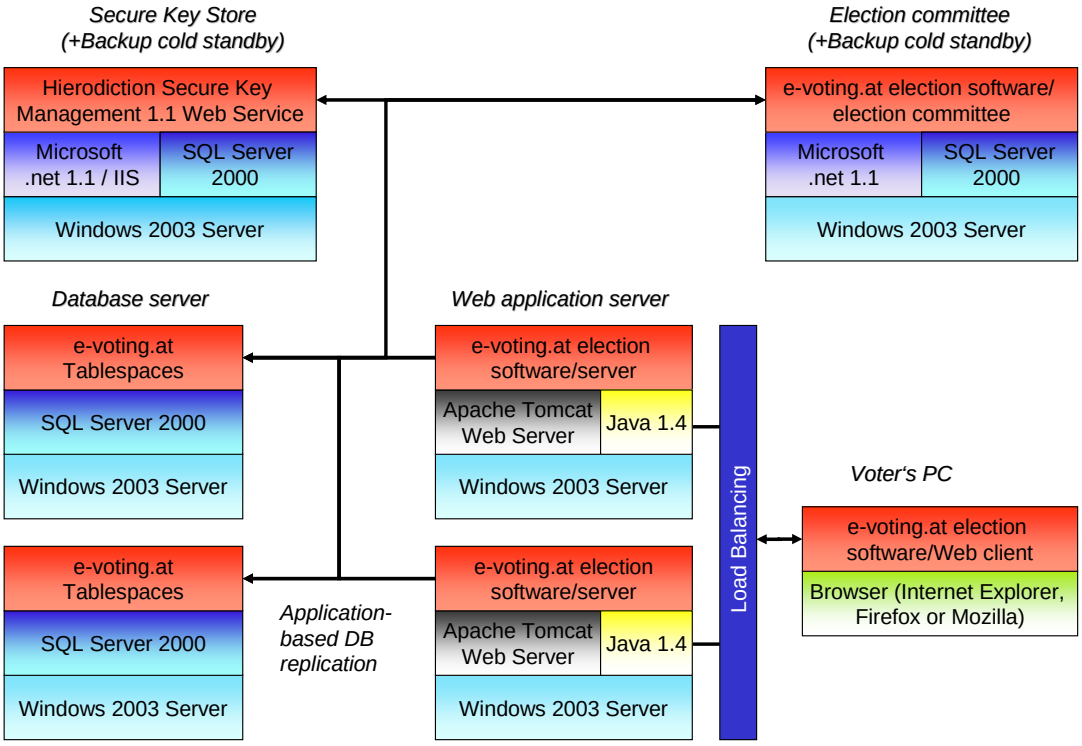

Figure 5: Implementation (for trademark information see References)

Productive Environment

Figure 5 shows the implementation overview. The following *logical* building blocks can be distinguished:

A. Self registration for the test (R1 and R2 in Fig. 1).

B. The electoral roll function, where users obtain a voting token (all remaining steps in Fig. 1 except for Step R5b); the prototype as such uses login-based authentication, which was deactivated as there was no electoral roll in this unofficial test. In fact, everybody who passed the self-registration in Building Block A was forwarded as authenticated to the election system itself.

C. The verifier (Step R5b).

D. The ballot box (all steps in Fig. 3).

E. The election committee function (all steps in Figs. 2 and 4 except for P2, 3, 4a,b and C2-4).

F. The Key Store (the steps excepted in E).

They can be assigned to the physical building blocks as follows:

A to D in the Web application server was duplicated in a load balancing configuration and used the (also duplicated) database server. The hardware used was standard single-processor Fujitsu-Siemens Primergy servers with 2 GB RAM.

E was implemented on the Election Committee server in Fig. 5, for which a Primergy double processor system with 2 GB RAM was used.

F was implemented on the Secure Key Store server in Fig. 5, which was the only server not hosted by Wiener Zeitung, but was made accessible via a Web service by the software vendor. The hardware used was a Dell single processor Dual Core server with 1 GB RAM.

A – C were based on a prototype developed at WU in 2005 in a research project for implementing and

verifying the cryptographic parts of the processes depicted in Figs. 1 and 3. As can be seen in Fig. 5, it is based on Java 1.4 and Apache Tomcat.

F is a commercial standard system implemented as a Microsoft IIS Web Service and E was specifically developed for this project using the Microsoft .net framework and SQL Server.

At the beginning of the project, major concerns existed in regards to whether the cryptographic implementations, particularly that of RSA and AES, would work smoothly together. The Java prototype stores cryptographic keys in plain hexadecimal format, whereas Microsoft .net uses its XML-based Crypto Service Provider format, which has a completely different numerical representation of the keys. In addition, to use the BigInteger classes that were used in the Java prototype for the cryptographic computations[3], Microsoft's BigInteger implementation (part of its Java implementation) had to be used, where possible performance implications had to be considered.

As it turned out, neither of these points was an issue and even though this was not part of the intended project scope, through this project it can be seen that Java-based cryptography and .net applications are able to work smoothly together and with good performance.

Development Environment

The development environment for the prototype was Netbeans 4 with its bundled Tomcat Web Server and Windows XP SP2. The primary test environment before deployment was Windows 2003 Server SP1, Apache Tomcat and Java RTE 1.4.2, hence, the same environment as the productive system.

Test Environment E-Voting Prototype

The system was developed and functionally tested under Java 1.4.2 and Internet Explorer. However, also the current versions of Firefox and Mozilla were tested. Only the Opera Browser experienced major issues in rendering the Java interface. The issues could not be resolved completely and hence Opera was not added to the list of supported Browsers.

In spite of complete downward compatibility according to documentation, the ballot sheet was not rendered correctly under Java 1.5 and some modifications had to be made to the coding in order to render it correctly for this test.

We were not sure, how many voters to expect, hence, a stress test was conducted. Robots were constructed and installed implementing repetitive loops interacting with the test server. Apart from several smaller elections two major elections with 10000 and 30000 participants, respectively, were simulated. The system environment for the "robot" PCs was Windows XP SP2 and Windows 2000 SP4.

Test Environment Key Store and Election Committee Server

The Key Server used was a standard product that can be installed on Windows XP2 IIS 5.1 as well as Windows 2003 Server IIS 6. Due to the small number of interactions in the test, it was decided to use an XP server. The Web Service implementing the Key Store was not available locally at the premises of Wiener Zeitung but was accessed remotely at the vendor's premises.

The most critical function of the election committee function was to open and count the ballot once the election committee members provided their private keys. For each vote, three RSA keys had to be opened, with the key length being 512, 784 and 1024 bit, resp.

Two stress tests were conducted:

A test with 10000 votes, where opening took 22 minutes (0.132 seconds per vote);

A test with 30000 votes, where opening took 1 hour and 7 minutes (0.134 seconds per vote).

This would indicate that opening time increments linearly; operating parameters during opening were observed, at no stage, RAM became an issue and RAM usage remained well below the space physically available (2 GB).

[3] Needed for all non-standard cryptographic functions, such as blind signatures and some hash functions.

5 Usability Test

This section provides the results of a usability test conducted with 16 test persons.

The target group for the usability test were standard users with basic Internet knowledge, especially people who are not very experienced with computers. 16 persons with basic or moderate computer skills were randomly chosen. As Table 1 shows, disproportional emphasis was put on members of the higher age groups. There was an almost even distribution among men and women.

The purpose of the tests was to yield information on how to improve the website as well as the e-voting software. The main focus was to make it as easy and as simple as possible. The tests were conducted at an early stage of the project and several modifications were made to the software as a result of the observations.

Setup

The tests were conducted under Windows XP, 10 test persons used Internet Explorer, 6 persons the Firefox browser.

Before the test started the subject was given a short explanation about e-voting and where he or she can find the web site[4]. There was no interaction between observer and test person during the test, the observer limited himself to recording the interactions of the test person. All steps and problems were noted on a checklist.

1. The usage of the website and the help content for the registration process.
2. The registration process
3. The help content for the voting process
4. The voting process

Particular emphasis was put on providing adequate help and background information on the test; the Web page is still available at www.e-voting2006.at. It consisted of help information for registration and vote casting including screen cam shows for both steps and technical background information relating the steps in the user interface to the technical process. Table 2 however shows that the users took hardly any notice of the help information.

The obvious solution is to integrate help information "just-in-time" when it is needed in the process, however, this may conflict with the aim of minimizing the user interface and user interaction when the person votes.

Registration

Another problem was, that 3 test persons mistook the screenshots in the original help page with the real software and tried to enter data in the first screenshot (the self registration screen). This was solved in the revised version by clearly labelling the screenshots as such.

8 out of 16 test persons experienced difficulties in finding the link to the Web page in order to start the e-voting software. As a result, the starting link was disproportionately emphasised in the revised version.

A Java applet has very limited access privileges to local resources, unless it is digitally signed by the issuer. For budgetary reasons and since this was only a test, a self-signed applet was used instead of a digital signature that could be traced back to a root certificate by the respective browser. In the case of a self-signed applet, a security warning pop-up screen appears asking the user, whether he or she wants to reject, accept or always accept the signature from that issuer. 4 users who rejected the signature had to restart the applet, from the referring page, which worked in all cases.

The address screen for self registration was understood by all test persons (Step R1 in Figure 1).

The next screen was to enter and confirm the password for the encoding of the voting card and to indicate whether the recovery option (see Section 2) was selected. 2 users had problems with the minimum password length (8 characters) required. In response to this issue, additional information was included on the password entry screen (Step R2 in Figure 1) in the revised version. The same applied to the recovery function, which was not understood by 4 test persons.

[4] For easy readability, the text generally uses the male form.

The file dialogue in the registration process (R4) was understood by all test persons.

A main issue seems to have been the long response time for the token generation (typically 5 to 8 seconds depending on server load and local processing power of the user's PC). Most test persons started to impatiently click on the screen, hence a waiting message was included in the revised version used.

In order to have a more realistic setting, a break was made after registration.

Voting

4 Persons had issues with entering/remembering the password in Step V1. To facilitate user handling, failure to correctly parse the voting token in V3 resulted in positioning the user once again in the password entry screen in V1 (see Fig. 3).

It was generally expected that finding the voting token in the file system in Step V2 would be a major issue for most inexperienced users. The problem seemed particularly stringent, as helpdesk staff can only help the voter in finding the token, if the voter remembers its name. However, only one test person had problems finding the voting token. Also during the test itself, this was not an issue. The fact that users didn't have any problems in finding their voting cards is probably one of the most interesting results of this test. It shows that also inexperienced computer users can handle voting token files and use them appropriately to cast a vote.

4 test persons experienced issues with the ballot sheet: Some test users tried to enter several choices in the screen form, which was, of course, prevented by the screen control. The ballot sheets used in the non-binding tests in 2003 and 2004 were rather straightforward: In 2003 it listed the parties for the student elections at WU and in 2004 the names of the two candidates for the Presidential Election. This time, the ballot sheets suggested some improvements for casting votes from abroad (see Fig. 6). For each main option, except for "Keine davon" (none of these), which corresponds to a blank vote, some sub-options were also available. Of course, the sub-option chosen had to correspond to the selected main option. The user was therefore requested to pass through a two-stage selection process. 4 test persons had problems understanding this mechanism and tried to select sub-options for a main option that had not been selected or to select more than one option.

Age	%
15-19	0%
20-29	19%
30-39	6%
40-49	13%
50-59	56%
>60	6%

Education	%
Compulsory school	13%
Apprenticeship	38%
Vocational school	0%
Secondary school	31%
University	19%

Gender	%
Women	56%
Men	44%

Table 1: Demographic factors in sample

Help page	%	Pers.
Read helppages	0%	0
Short look	63%	10
Did not use help page	38%	6

Technical explanation	%	Pers.
Read technical explanation	6%	1
Short look	31%	5
Did not read	63%	10

Table 2: Usage of help information

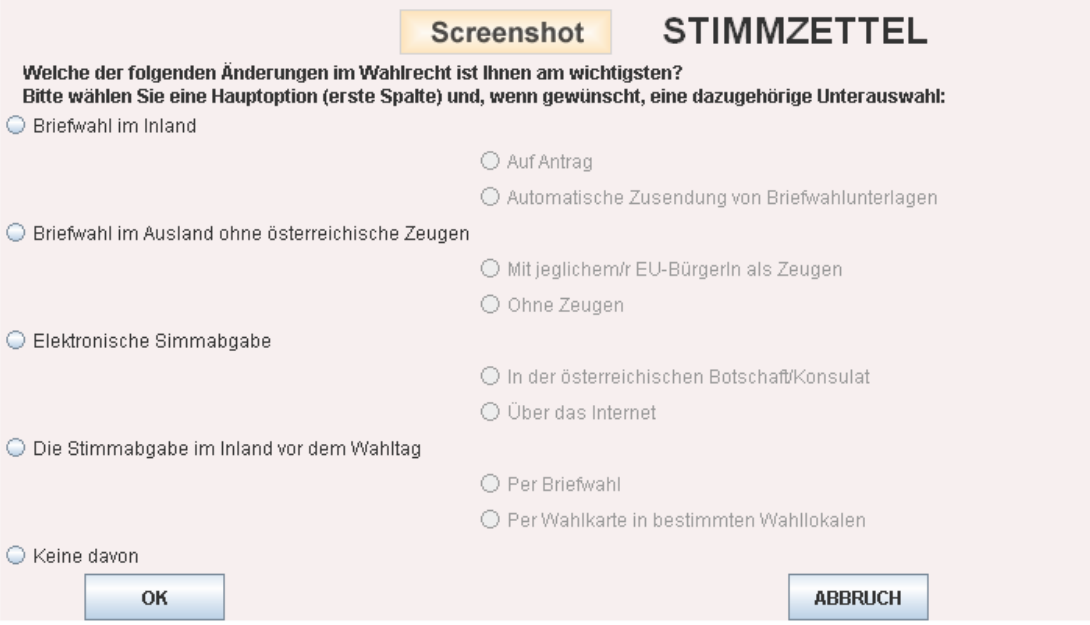

Figure 6: Ballot sheet

6 The Test

Time Frame and Helpdesk Requests

The registration phase took place between 25th September 0:00 am and 11th October 12:00 pm. Vote casting took place between 12th October 0:00 am and 24th October 5:00 pm. Email support was available during registration and a helpdesk person for telephone support from 9 am to 6 pm CET during vote casting. 293 persons registered and 148 persons actually cast a vote. Austrians abroad were the main target group and obviously it had been difficult to approach this group in the given time frame. Nevertheless valuable insights were gained from the feedback of users that, unlike WU students in previous tests, were not completely familiar with the use of computers.

All told, Helpdesk logged 19 requests:

2 users forgot their passwords: In this case Helpdesk could not help in any way, as it is the very intention of the mechanism to stop anybody other than the voter himself to gain access to the voting token. In 2 other cases, it could not be decided whether the token file had been damaged or the voter had forgotten the password.

4 persons tried to view the token by double clicking on it, which, of course, did not work, as there is no application linked to it. In fact, the token is a simple text file containing the result of the symmetric encryption process in text form. It can be viewed with any text editor, such as Windows Notepad. The wording used for the voting token was “elektronische Wahlkarte” (electronic voting card), which may have led users to believe that, if opened, it would display some sort of text or an emulation of a paper-based voting card. In fact, the voting card was a signed bitstring that only fulfilled the *function* of a voting card. This obviously has to be better communicated to the end user.

One user experienced problems with Java 1.4.1. The applet required 1.4.2, which was probably not communicated clearly enough in the help information. The user was guided by the Helpdesk to download the current Java version 1.5 and was able to proceed.

Nobody else lodged a request that would indicate that Java had not been installed on the participant’s PC. One may only hypothise about the reasons, one reason may be that in many cases pre-configured PCs sold via consumer channels actually do have a current Java version installed and users are not required to install Java themselves. Users who build the software environment on their PC themselves, on the other hand, would have been able to follow the instructions on the help pages to download and install the free Java run time environment.

One user was confused that a self-signed applet was used, a procedure which was chosen for cost reasons (see above). In a real election, any such system would obviously use a valid signature that can be traced back to a root-certificate.

After the vote had been cast, the user was re-routed to a questionnaire (see below), one person did not receive the questionnaire and logged a Helpdesk request.

The remaining Helpdesk requests were suggestions for improvement, most concerning the use of a signed applet.

An Unsollicited Attack on the System

The prototype developed at WU was used in this test with two add-ons: (i) The self registration and (ii) the questionnaire (see below).

Due to an oversight by the programming team at WU, there was an issue in the deployment of the self-registration add-on to the prototype. Some parts of the server functionality of the self-registration were in fact configured to be delivered to the voter together with the applet. In an unsolicited attack on the system, Mr Leitold from a-sit (www.a-sit.at) attacked the prototype system and due to the aforementioned issue was able to compromise the add-on. The attack as such was immediately noticed by Wiener Zeitung IT staff, which triggered a security check of its infrastructure. The deployment issue in the configuration of the add-on, however, remained unnoticed. Unfortunately, Mr. Leitold did not disclose his findings to the project team, otherwise the self-registration add-on could have been easily redeployed.

However, as was established later beyond doubt he was not able to successfully penetrate or compromise the election as such. The processes employed did not enable anybody to forge or modify votes or to

break voter anonymity and no such claims were actually made. This attack proves that the processes implemented in order to maintain voter security and anonymity work even when one is able to directly access information stored in the database itself. It should be once again mentioned that this system is designed not just to secure an election protecting it from hackers but also from the system administration, who may wish to manipulate the election. The System Administration has complete access to the database storing the voter information, hence the necessity for highly developed encryption and decryption processes to protect the integrity of the election.

As mentioned, the primary focus of the test was on usability and the acceptance of a two-stage voting process, a security test was not conducted and not within the project scope. Nevertheless this unsolicited attack showed that the processes employed provide a strong protection from fraudulent attacks designed to compromise an election. These technical aspects are to be explored in more details in separate publications. To follow the publications issued by the project team, please consult www.e-voting.at.

Result of the Ballot

Table 3 depicts the result of the ballot. Not very surprisingly, electronic voting was the modification to the election processes of choice for the participants in the e-voting test. Of course, the result has a strong bias, however, some conclusions may be drawn nevertheless:

"Stationary" e-voting via terminals is not particularly popular even among those who do favour e-voting in general. This corroborates the results of a Working Group on e-voting in the Ministry of the Interior [BMI05], which did not recommend terminal-based e-voting in polling stations.

Interestingly, the idea of advance voting for residents in Austria, who are not in the country when the elections takes place, seems to have some support, where even the possibility of advance voting in selected polling stations is the preference of choice for the majority of those in favour of advance voting.

Option/sub-option	Number	Percent
Postal voting in Austria	6	4%
No sub-option chosen	0	
Upon request	4	
Automatic sending of postal voting documents	2	
Postal voting abroad without Austrian witness	12	8,1%
No sub-option chosen	0	
Any EU citizen as witness	6	
No witness	6	
Electronic voting	118	79,7%
No sub-option chosen	2	
In Austrian embassy /consulate	2	
Via the Internet	114	
Advance voting	8	5,4%
No sub-option chosen	1	
Per postal voting	2	
In selected polling stations	5	
None of the aforementioned	4	2,7%

Table 3: Result of the ballot

Recovery Function

For the first time, a recovery function for the voting token was offered (see Section 2). By default, the option was *de-selected* and one of the prime questions of this test was, whether the recovery function would be understood, trusted and actively used.

153 out of 297 participants selected the recovery, which is 51,5%. This shows that the recovery function meets a real need and is accepted by the users of an e-voting system. Interestingly, none of the users during the test seem to have had any difficulties understanding and using the recovery function, as there were no requests in this regard.

Two persons actually used the recovery function, in that the encoded voting token was sent to their specified email addresses. Also in two cases, where it was not clear, whether the password had been forgotten or the token file was corrupted, the token

was resent (both had used the recovery option). In a paper-based mail voting process, the vote would have lost his possibility to vote. The recovery function in electronic voting enables the voter to securely store and, if need be, request the voting token.

Results of Questionnaire

After they had cast their vote, participants were re-routed to a one-page questionnaire about their user experience. Table 4 shows the questions (translated into English) and the results. In 4 questions, users could indicate their support for a statement on the usual 5 point Likert scale (1 indicating strong support), the last one was a yes/no question.

85% of the respondents indicated strong or very strong support that the registration was easy to use; the corresponding value was 83% for vote casting. This confirmed the result of the usability test, where the majority of the test persons could work through the dialogues without a problem. Also, it reflects the modifications made in response to usability difficulties encountered in the tests. Only 6% of the respondents (strongly) disagreed that the prototype was easy to use. This seems to be remarkable for an application, participants cannot have encountered before, and hence would not have learnt its mechanisms in similar applications.

In the literature it is increasingly recognized that using an anonymised voting token is the prime way to technically guarantee voter anonymity (see Section 3). However, this guarantee comes at a price: Two interactions are necessary for the user to cast a vote (two-stage process). The question is, whether this would be accepted.

In a remarkable result, 90% of the respondents (strongly) supported the idea of a two-stage voting process. This may also reflect the fact that voters have already “learnt” the two-stage postal voting process with application and voting itself. That was also the prime reason for the wording “electronic voting card” for the token.

68% of the respondents expressed (strong) support in the system and its ability to correctly conduct the election and to protect voter anonymity. 19% were neutral and 12% obviously did not trust the system. The only way to counter these concerns is transparency about how the systems works and the ability of the election committee/s to check the system at all times and in all stages of an election

Finally, participants were asked whether they had used the help pages provided with the prototype. The result corroborated the observations of the usability lab in that the majority did not (even briefly) consult the help information.

Question (Engl. translation)	1	2	3	4	5	Sum
The registration for the e-voting test was easy to use	51	46	10	5	2	114
	45%	40%	9%	4%	2%	100%
The vote casting was easy to user	62	33	12	6	1	114
	54%	29%	11%	5%	1%	100%
Two actions are necessary to cast a vote, to register and to vote. I am prepared to accept this if it is necessary to effectively protect voter anonymity	94	9	4	3	4	114
	82%	8%	4%	3%	4%	100%
I am confident that the system correctly records my ballot and protects my voter anonymity	48	30	22	9	5	114
	42%	26%	19%	8%	4%	100%
	Yes	No	Sum			
I have used the help pages and the information on how the system works	48	66	114			
	42%	58%	100%			

Table 4: Result of questionnaire

7 Lessons Learnt

Functionality

The **recovery** function was used and widely accepted and obviously meets a clear need.

The elaborate **help page** provided was generally not used (which should not lead one to the conclusion that it would be unnecessary, however it cannot be a panacea). Rather, help information has to be included in the voting process to be provided to the user "just in time". This however conflicts with a parsimonious interface to minimize various risks, from undue influence on the voter's decision, via compatibility issues, pop-up blocking to various technical vulnerabilities. To reconcile these two aims will be a major challenge.

A **two-stage voting process** is acceptable to users, if the reasons are explained. If it is necessary to preserve voter anonymity, users tend to accept the additional effort.

The mechanics of the voting token as experienced by the users has to be more transparent to the user, which was clearly shown by the attempts to extract some sort of textual information from the token.

Usability

The vast majority found the existing prototype easy to use; this in spite of the multiple stages to be performed: register (authenticate yourself in a real system), specify password and file name (registration) and password entry, file specification and filling in the ballot sheet (voting stage). Hence, the main principals of **user guidance** and of the logic employed in this test can be used as a foundation for further work.

In regards to the **sub- or preferential options**, some test persons in the usability lab experienced difficulties in handling the options. However, it is interesting to note that most votes actually cast in the test contained a sub-option (whose indication was optional). Therefore one is lead to conclude that the preferential mechanism was understood in the end. Nevertheless, user guidance by the system to prevent invalid votes is clearly indicated.

Two issues that were expected by the project team did not manifest:

Also inexperienced users seem to be able to handle **token files** ("voting card"), to find them again and associate the correct password with it.

The **Java download** and Java version triggered a single helpdesk request. This would indicate that (i) in spite of the issues with Java and the Windows platform, a Java runtime environment is a reality also on Windows PCs and (ii) that users can be comfortably and securely guided to download the run time environment if needed.

Transparency

Results of the questionnaire indicate that an electronic voting system will only be universally trusted if all processes are completely transparent. System certification and software signing by independent authorities will play a major role in this context.

References

[BMI05] Bericht der Innerministeriellen Arbeitsgruppe über die Möglichkeiten von e-voting in Österreich (retrievable at http://evoting.at/main.php?ID=93)

[Cha82] Chaum, D.: Blind Signatures for Untraceable Payments in Advances in Cryptology Proceedings of Crypto 82, D. Chaum, R.L. Rivest, & A.T. Sherman (Eds.) pp. 199-203.

[CC06] ISO IEC 15408, Common Criteria for Information Technology Security Evaluation Version 3.1, 2006

[CoE2004] Committee of Ministers' recommendations on e-voting (Rec(2004)11), Strasbourg, 2004 (can be retrieved at http://www.coe.int/t/e/integrated_projects/democracy/)

[WU2004] Prosser, A., Kofler, R., Krimmer, R., Unger, M.. 2004. E-Voting Wahltest zur Bundespräsidentschaftswahl 2004. Working Paper 01/2004 des Institut für Informationsverarbeitung und -wirtschaft

[PKK03] Prosser, A., Kofler, R., Krimmer, R.: Implementing an Internet-based Voting System for Public Elections - Project Experience. Proceedings of ICEIS 2003, Setubal, 2003.

[Sch01] Schneier, B.: Fundamentals of Cryptography; Kluwer, Boston, 2001